

Position Estimation Using a Stereo Camera as Part of the Perception System in a Formula Student Car

by

Eloi Bové Canals

Advisor: Josep R. Casas

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

In partial fulfilment

of the requirements for the degree in TELECOMMUNICATIONS TECHNOLOGIES AND SERVICES ENGINEERING

Barcelona, June 2019

Abstract

This thesis presents a part of the implementation of the perception system in an autonomous Formula Student vehicle. More precisely, it develops two different pipelines to process the data from the two main sensors of the vehicle: a LiDAR and a stereo camera.

The first, a stereo camera system which is based on two monocular cameras, provides traffic cone position estimations based on the detections made by a convolutional neural network. These positions are obtained by using a self-designed stereo processing algorithm, based on 2D-3D position estimates and keypoint extraction and matching.

The second is a sensor fusion system that first registers both sensors based on an extrinsic calibration system that has been implemented. Then, it exploits the neural network detection from the stereo system to project the LiDAR point cloud onto the image, obtaining a balance between accurate detection and position estimation.

These two systems are evaluated, compared and integrated into “Xaloc” – The Formula Student vehicle developed by the Driverless UPC team.

Resum

Aquesta tesi presenta una part de la implementació del sistema de percepció en un vehicle autònom de Formula Student. En concret, es desenvolupen dos sistemes diferents per processar les dades dels dos principals sensors del vehicle: un LiDAR i una càmera estèreo.

El sistema de càmera estèreo es basa en dues càmeres monoculars, i proporciona estimacions de les posicions dels cons de trànsit que delimiten la pista basades en les deteccions fetes amb una xarxa neuronal convolucional. Aquestes posicions s'obtenen mitjançant un algoritme de processament d'estèreo propi, basat en estimacions de posició 2D-3D i en extracció i correspondència de "keypoints".

El segon és un sistema de fusió de sensors que registra els dos sensors en base a un sistema de calibratge extrínsec que s'ha implementat. A continuació, fa servir les deteccions de la xarxa neuronal del sistema estèreo per projectar el núvol de punts LiDAR a la imatge, obtenint un equilibri entre una bona detecció en imatge i la precisió del núvol de punts LiDAR.

Aquests dos sistemes són avaluats, comparats i integrats al "Xaloc" – el vehicle sense conductor de l'equip de Formula Student Driverless UPC –.

Resumen

Esta tesis presenta una parte de la implementación del sistema de percepción en un vehículo autónomo de Formula Student. Concretamente, se desarrollan dos sistemas diferentes para el procesado de datos de los dos sensores principales del vehículo: un LiDAR y una cámara estéreo.

El sistema de cámara estéreo se basa en dos cámaras monoculares y proporciona estimaciones de la posición de los conos de tráfico que delimitan la pista en base a las detecciones realizadas por una red neuronal convolucional. Estas posiciones se obtienen mediante el uso de un algoritmo de procesamiento estéreo de diseño propio, basado en estimaciones de posición 2D-3D y en extracción y correspondencia de “keypoints”.

El segundo es un sistema de fusión de sensores que primero registra ambos sensores basándose en un sistema de calibración extrínseco que se ha implementado. Luego, usa la detección hecha con la red neuronal del sistema estéreo para proyectar la nube de puntos LiDAR en la imagen, obteniendo lo mejor de cada sensor: una detección robusta y una estimación de posición muy precisa.

Estos dos sistemas se evalúan, comparan e integran en “Xaloc” – el vehículo sin conductor del equipo de Formula Student Driverless UPC.

Acknowledgements

This project has been developed as part of the Driverless UPC project of building an autonomous race car. An acknowledgment has to be made to the team itself, which is formed by 21 engineering students from multiple disciplines, and through the course of this project, a lot of collaboration has been established with all the team members.

An acknowledgement also to the Open Source community for developing and sharing tools such as ROS, OpenCV, and PCL, which were fundamental for the creation of this thesis. Special thanks also to the Formula Student Driverless community for sharing datasets, documents, etc.

Also, a special acknowledgment has to be given to the Team Leaders, Sergi Catalán and Monica Pérez for their guidance, as well as the Tutor of this thesis, Dr Josep R. Casas for his wise counselling. Also to Dr Albert Aguasca for his crucial support.

Last but not least, the work developed in this thesis is highly influenced by the Perception department. This project wouldn't have been possible without their mutual help and support.

Revision history and approval record

Revision	Date	Purpose
0	01/05/2019	Document creation
1	24/06/2019	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Eloi Bové Canals	eloibove@gmail.com
Josep R. Casas	Josep.ramon.casas@upc.edu

Written by:		Reviewed and approved by:	
Date	23/06/2019	Date	24/06/2019
Name	Eloi Bové Canals	Name	Josep R. Casas
Position	Project Author	Position	Project Supervisor

Table of contents

1	Introduction	11
1.1.	Formula Student: Driverless UPC	11
1.1.1.	The competition	11
1.1.2.	The team	12
1.2.	Objectives	13
1.3.	Requirements and specifications	14
1.3.1.	Requirements	14
1.3.2.	Specifications	14
1.4.	Methods and procedures	14
1.5.	Work plan	15
2	State of the art	17
2.1.	Formula Student	17
2.1.1.	Introduction to FSD	17
2.1.2.	AMZ Racing	17
2.1.3.	TUfast Racing	18
3	Methodology	20
3.1.	Software Architecture	20
3.1.1.	Introduction to ROS: basic architecture	20
3.1.2.	Rosbag: record and playback	20
3.1.3.	Rviz: powerful visualization	21
3.1.4.	Nodelets: zero-copy transport	22
3.1.5.	Message filters: synchronization between topics	22
3.1.6.	OpenCV: optimized computer vision	23
3.1.7.	PCL: cutting edge point cloud processing	23
4	Development	24
4.1.	Physical setup	24
4.1.1.	Sensors used	24
4.1.2.	Processing units	25
4.1.3.	Support and casing	26
4.2.	Perception Systems	28
4.3.	Stereo pipeline	28
4.3.1.	Introduction	28
4.3.2.	Camera calibration	29
4.3.3.	Image acquisition and trigger	31
4.3.4.	Synchronization and rectification	32

4.3.5.	CNN Cone detection	33
4.3.6.	Stereo processing	34
4.4.	Sensor fusion	36
4.4.1.	Introduction	36
4.4.2.	Calibration	37
4.4.3.	ROS implementation and point cloud projection	38
5	Results	40
5.1.	Stereo pipeline results	40
5.2.	Sensor fusion pipeline results	42
6	Budget	44
7	Conclusions and future development:	45

List of Figures

Each figure in the thesis must be listed in the “List of Figures” and each must be given a page number for its easy location.

Figure 1. Dynamic and static events.....	11
Figure 2.Using the cone colour to plan the trajectory.....	12
Figure 3. 2019 Driverless UPC race car: CAT12d - "Xaloc". The cameras can be seen mounted under the main hoop and the LiDAR sensor on the front wing	12
Figure 4. Autonomous design of the CAT12d	13
Figure 5. Final work plan	16
Figure 6. Vision from the perspective of “gotthard” 2018 AMZ Racing car [2]	17
Figure 7. Viewpoint from the left stereo camera (left) compared to the central mono camera (right).....	18
Figure 8. Triangulation of the distance based on the cone height.....	18
Figure 9. Data association based on GMM	19
Figure 10. ROS structure example.....	20
Figure 11.Timeline of rosbag shown on rqt_bag.....	21
Figure 12. 3D model of a car and its environment acquired by sensors in rviz.....	21
Figure 13. ApproximateTime sync with jitter [5]	23
Figure 14. ApproximateTime sync with different frequencies [5]	23
Figure 15. Trade-off between the characteristics of the used sensors	24
Figure 16. Velodyne VLP-32C (left) and 2x DFK33UX252 (right)	25
Figure 17. Range and FOV of the developed perception systems	25
Figure 18. NVIDIA Jetson TX2 (left) Cincoze DX-1000 (right)	26
Figure 19. Envelope to mount sensors.....	26
Figure 20. Stereo camera and LiDAR supports.....	26
Figure 21. Connection diagram inside the PU box.....	27
Figure 22. Perception block diagram.....	28
Figure 23. Stereo pipeline diagram.....	29
Figure 24. Barrel distortion (left) and pincushion distortion (right).....	30
Figure 25. Extrinsic transform between left and right imagers.....	30
Figure 26. Reprojection error as the distance between the detected and reprojected points.....	31
Figure 27. Image rectification. Epipolar lines are horizontal on the rectified plane [8]	31
Figure 28. Free-running cameras timestamps	32
Figure 29. Hardware triggered cameras timestamps	32
Figure 30. Inverse mapping to obtain the lookup table [13]	33
Figure 31. Epipolar lines in a pair of rectified images	33
Figure 32. Colour and physical characteristics of the cones	34
Figure 33. First training of the neural network.....	34
Figure 34. 2D-3D correspondences of a face filter to insert an augmented 3D object.....	35
Figure 35. Bounding box propagation scheme (left) and feature matching (right)	35
Figure 36. Processing of the LiDAR and Stereo point clouds.....	37
Figure 37. Sensor fusion ROS diagram	38
Figure 38. Sensor fusion: project the point cloud to the image plane and detection using the CNN	39
Figure 39. Estimated cone positions in comparison to the ground truth (left) and estimated distance boxplot (right).....	40
Figure 40. Theoretical vs empirical RMSE	41

Figure 41. Timings of the stereo pipeline.....	41
Figure 42. Two different setups of Stereo and LiDAR point clouds superimposed.....	42
Figure 43. Estimated cone positions vs ground truth (left). Sensor fusion distance estimation statistics (right).....	43
Figure 44. Track limits and path planning using the sensor fusion	43

List of Tables:

Each table in the thesis must be listed in the “List of Tables” and each must be given a page number for its easy location.

Table 1. Milestones.....	15
Table 2. Budget.....	44

1 INTRODUCTION

1.1. FORMULA STUDENT: DRIVERLESS UPC

1.1.1. The competition

The purpose of this project emerges from the Formula Student Driverless challenge [1], an engineering competition that gathers teams of students from all around the world to compete in a series of events or tests. The overall goal of the teams is to design and build an autonomous race car that is able to overcome all the challenges that this competition entails.

The events that this competition include are divided into two groups: the static and the dynamic events. The three static events consist on Business Plan, Cost and Manufacturing and Engineering Design, which are documents and presentations related to marketing, finances and the overall design choices of the vehicle. These events represent roughly half of the competition points. The dynamic events, however, focus on testing and putting to the limit the abilities of the car itself. Each event is designed to take into account different features of the vehicle, such as the acceleration, the behaviour in closed curves, the endurance, etc. However, the most demanding dynamic events are the *Trackdrive*, which consists in completing 10 laps in a closed circuit, from which the car doesn't have any previous knowledge, within the shortest time possible, and the *Autocross*, which consists in completing a single lap in the same circuit with the same rules.

The combination of these two types of events grant that, as the FSG says, "*The competition is not won solely by the team with the fastest car, but rather by the team with the best overall package of construction, performance, and financial and sales planning*".

The driverless dynamic events are based on a controlled environment. This means that the track that the car needs to follow is marked with differently coloured cones, blue on the left and yellow on the right. This helps the cars to determine the trajectory that needs to be followed, as can be seen in Figure 2.

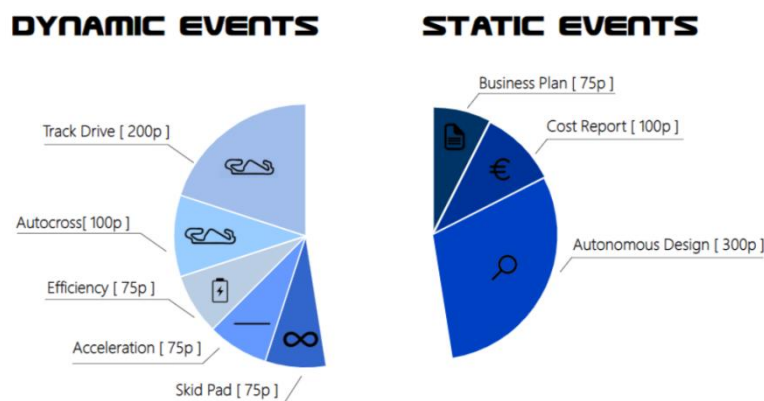


Figure 1. Dynamic and static events

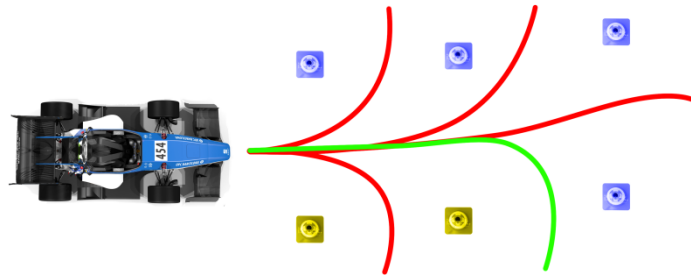


Figure 2. Using the cone colour to plan the trajectory

1.1.2. The team

The Driverless UPC team was born as a part of ETSEIB Motorsport, a team that has been competing in the Formula Student Electric competition for 8 years. Driverless UPC is participating for the first time in the Formula Student Driverless with its new prototype: the CAT12d. This vehicle was built using the body and dynamics of the CAT10e and the electronics and power train of the CAT11e, which were the former electric prototypes developed by ETSEIB Motorsport, while designing and implementing the autonomous system and all the needed adaptations and modifications to the already existing parts.

The team is organized in sections, which are Perception (PER), Estimation and Control (E&C) and Hardware (HW). These are organised to build the autonomous system of the car and also its adaptation.



Figure 3. 2019 Driverless UPC race car: CAT12d - "Xaloc". The cameras can be seen mounted under the main hoop and the LiDAR sensor on the front wing

The autonomous system of the vehicle is based on the diagram from Figure 4. The perception of the vehicle relies on two main sensors, the stereo camera and the LiDAR. These sensors perceive and locate the cones on the track. Then, the detections are fused to create a virtual map of the environment, fusing the data from perception, the localization and the speed information. The following step is the trajectory planning, which defines the path that the car needs to follow. Once this path is defined, the control algorithms generate and send the commands to the motors so that they can move according to the planned trajectory.

AUTONOMOUS DESIGN

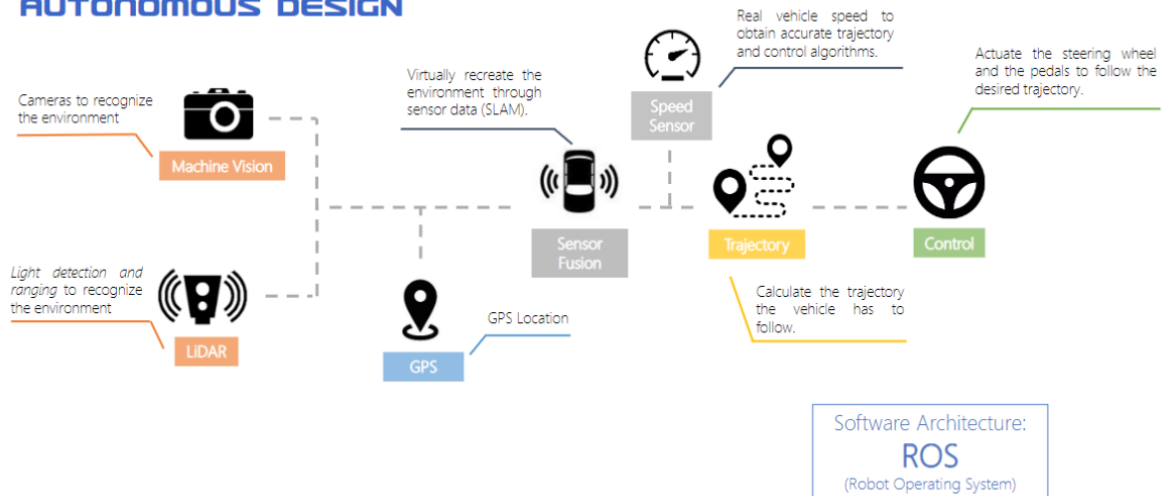


Figure 4. Autonomous design of the CAT12d

1.2. OBJECTIVES

The objectives of this project emerge from the Driverless UPC team objectives. Being a first-year team, these objectives are mainly focused on reliability and robustness, leaving performance and optimization as secondary objectives. For that, the main objectives of the team are getting good results in the static events, and to build a reliable and robust car so that it is able to finish all the dynamic events.

Regarding the Perception section, the main objective is to provide accurate positions to the Estimation and Control department. The developed systems must be able to overcome a sensor failure, having reliable alternatives to grant that the car will be able to run under such circumstances.

The aim of this work is to develop a stereo camera system that is able to estimate the 3D position of the detected cones. Also, in parallel, implement an extrinsic calibration algorithm for the stereo camera and the LiDAR sensor, so that their data can be compared and/or fused.

These systems need to be carefully designed so that the car can complete the most demanding challenges of the FSD competition: the *Trackdrive* and the *Autocross* events. These events are based on the same layout, a closed loop with the following guidelines:

- Straights: no longer than 80m
- Constant turns: up to 50m diameter
- Hairpin turns: minimum of 9m of outside diameter
- Minimum track width: 3m
- Maximum longitudinal distance between cones: 5m
- Miscellaneous: chicanes, multiple turns, decreasing radius turns, etc.

1.3. REQUIREMENTS AND SPECIFICATIONS

1.3.1. Requirements

The requirements for the stereo and the sensor fusion systems are the following:

- The pipelines need to be able to run in real-time, allowing the same thing for all the algorithms and processes that come after them: from the path planning to the control of the motors.
- There is a need for a good colour and distance estimation, to provide a good base for the Estimation and Control department.
- All the developed algorithms need to be in consonance with the requirements of the HW and the E&C department. This includes the characteristics of the data that is being transmitted, the transmission of the data itself, the operating characteristics of the sensors, etc.
- The setup needs to be resilient to vibrations, to withstand the conditions in which the vehicle needs to operate and not sacrifice accuracy for that.

1.3.2. Specifications

The specifications of the systems are described in this section.

- The range of the detections is specified by the E&C department. The algorithm that finds the track limits and the trajectory planning need, at least, three pairs of cones to obtain a good enough estimation of the track layout. This distance will vary depending on the track but, in the worst case, three cones with a separation of 5m results in a range of 15m.
- The horizontal field of view of the system should allow the vehicle to see enough cones in a closed curve.
- The error in the positions of the cones should be comparable to the error in depth of a commercial stereo camera, which can be modelled with an exponential curve depending on the depth.
- The maximum processing time of the perception pipeline, besides from working “in real-time”, is specified by the frequencies from the other devices and systems on the vehicle. The most limiting ones are the frequency of the LiDAR (10Hz) and the working frequency of the kinematic control of the vehicle, which can operate at 10Hz but lowering the frequency significantly lowers the maximum speed of the car. That means that the output frequency of the perception systems needs to be equal or higher than 10Hz.
- In fulfilment of the requirement of accuracy, the calibration phases of both sensors needs to be as accurate as possible. That means, at least, sub-pixel accuracy. That is because small errors in pixels can have great impact at long distances.
- The supports for the sensors need to be resilient to vibrations as they are the most susceptible to be affected by them, as the extrinsic calibrations highly depend on that.

1.4. METHODS AND PROCEDURES

This project, as stated before, is a part of the development of a greater project in the context of the Driverless UPC competing in the Formula Student Driverless competition. All the

contributions made by other team members and, to a bigger extent, by the members of the Perception department will be clearly stated in this section. Moreover, some parts of this project were developed by the Author in the context of the PAE subject, and are also clearly stated in the next paragraphs.

The CAD design of the cameras' casing, appearing in section 4.1.3 was conceived in collaboration with the Hardware department.

The convolutional neural network briefly explained in section 4.3.5 was developed by R. Aylagas, member of the Perception department as a part of his Final Degree Thesis.

The point cloud projection explained in the section 4.4.3 of the sensor fusion pipeline was developed by A. Roche, member of the Perception department as a part of his Final Degree Thesis, along with the whole LiDAR pipeline.

Finally, the first steps with the stereo calibration in MATLAB, and a simple and undeveloped stereo matching algorithm were part of the PAE project of the Author, along with a preliminary cone detection algorithm that went unused and substituted by the CNN in 4.3.5.

1.5. WORK PLAN

The final work plan for this project can be seen in Figure 5. This work plan does not present any major changes in relation to the original work plan presented in the Proposal & Work Plan document of this thesis.

WP	Short title	Milestone	Date
2	Full pipeline test	Visualization tool and LiDAR + Stereo algorithms	28/02/2019
3	Data acquisition	Camera driver ready	11/03/2019
4	Sensor fusion	Calibration software ready	16/04/2019
4	Car fully mounted	All the software systems ready	01/05/2019

Table 1. Milestones

Regarding the milestones, the first one was not accomplished, as it might have been too ambitious. Due to delays in the E&C and PER departments, the full pipeline test was delayed until May. However, internally in the perception department, the visualization of the LiDAR detections could be tested. The second and third milestones were indeed accomplished, in contrast to the fourth, which was delayed until June.

The biggest deviation from the work plan, besides the delays, was that the first one included the development of a neural network or a deep learning approach for the stereo matching algorithm, which didn't end up being developed. This is further detailed in section 4.3.6.

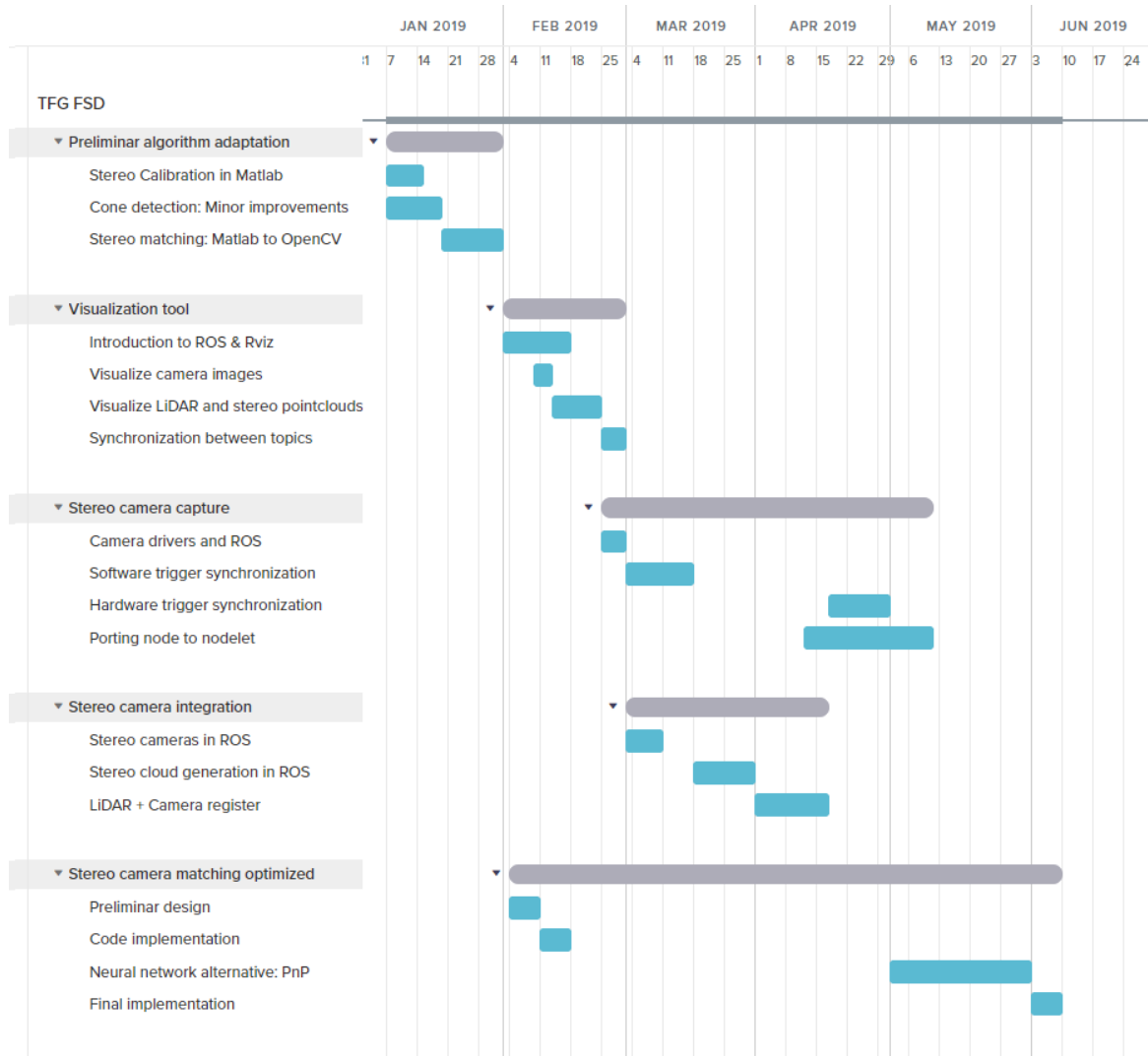


Figure 5. Final work plan

2 STATE OF THE ART

2.1. FORMULA STUDENT

2.1.1. Introduction to FSD

The Formula Student Driverless challenge started in 2017 and, since then, many teams have taken the autonomous driving challenge.

Each team can have very different approaches to design and implement their autonomous system. To start with, teams may choose from a wide variety of sensors. Most teams include some type of camera; either mono or stereo, many of the teams use LiDAR sensors and some of them even rely on RADAR. More importantly, the second part is to develop data processing algorithms that can reliably and accurately provide the positions of the cones that delimit the track, while also taking into account that these algorithms must be able to run in real time, be robust to failures, etc.

Two interesting approaches are the ones from Zurich ETH (AMZ Racing) and from Munich TU (Tufast Racing). These will be explained in the following sections.

2.1.2. AMZ Racing

AMZ Racing has been the winning team of the past two editions. Their car features a sensor layout formed by three cameras (two of them working as a stereo pair and one of them as a mono) and a LiDAR. The data that these sensors provide is processed in three different pipelines, mono, stereo and LiDAR, providing a lot of redundancy but also the same amount of robustness in the case that one of these pipelines were to fail.

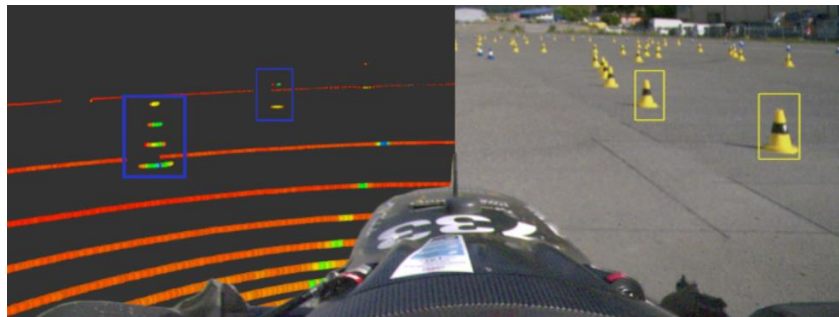


Figure 6. Vision from the perspective of "gotthard" 2018 AMZ Racing car [2]

The LiDAR pipeline is processed in three different steps. These are the pre-processing, the cone detection and the colour estimation. In the first step, the LiDAR scans are accumulated by using their velocity estimation of the vehicle. This process removes the artefacts that can occur in the point cloud when the vehicle is moving fast. Then the FOV is filtered to roughly 180 degrees and the ground is removed. This step leaves a filtered point cloud that allows the use of a simple clustering algorithm to detect the cones in the second step. Finally, each cluster is classified according to the cone colour using a colour estimation based on the infrared intensity perceived by the LiDAR.

The Stereo pipeline uses the state-of-the-art YOLOv2 [3] CNN to detect the cones in the images. This neural network can detect objects with a very good localization accuracy and a low misclassification rate, while running in real time. Then, with the knowledge of some cone

priors (size, width of the strip in the middle, etc.), the 3D pose of each detection is estimated by using a 2D-3D correspondence algorithm. This 3D pose is used to extrapolate the detection from one image to the other and then use keypoint extraction and matching to obtain the final position estimations for the cones.

Finally, the Mono pipeline uses the same structure as the Stereo pipeline, but uses the 3D pose estimation as the final position for the cone. This would cause no improvement in relation to the Stereo pipeline if it weren't for the mono camera optics that allows the mono pipeline to see much further than the stereo pipeline, as can be seen in Figure 7.



Figure 7. Viewpoint from the left stereo camera (left) compared to the central mono camera (right)

2.1.3. TUfast Racing

The team from Munich has achieved considerably good results in the last competitions (top 6) and they are distinguished by their simple yet effective approach.

The perception of their vehicle is based on a single monocular camera. The first step of their pipeline is the CNN cone detection, for which they also use the YOLOv2 convolutional neural network. Once the detections are done, the distance at which the cone is in relation to the car can be estimated by using the cones' height, as can be seen in Figure 8.

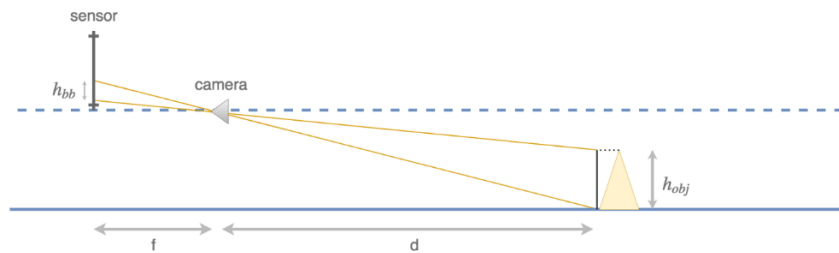


Figure 8. Triangulation of the distance based on the cone height

The simplicity of this approach is translated into a very low computational load, allowing the system to be running at a high frame rate. This fact is exploited by the data association algorithm, which fuses the multiple instances of each cone provided by the perception system based on a GMM or Gaussian Mixture Model for each cone.

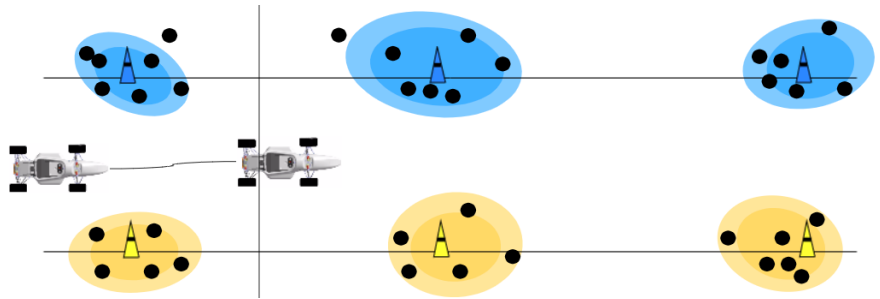


Figure 9. Data association based on GMM

3 METHODOLOGY

3.1. SOFTWARE ARCHITECTURE

3.1.1. Introduction to ROS: basic architecture

Robot Operating System [4], or ROS for short, is a set of software libraries and tools that are aimed to build robot applications. It is widely used in research and in the robotics industry, mainly because it integrates many state of the art algorithms and very powerful developer tools. Also, it is open source.

ROS bases its distributed architecture in different programs or nodes that normally communicate to each other using TCP/IP protocol. The information transmitted is organised in topics, which define the type of message that the nodes send or receive when they publish or subscribe to a certain topic. For example, the node for a camera driver publishes the obtained images at the topic `/camera/image_raw` and another node can subscribe to this topic and then publish some extracted features about the images.

These nodes can communicate thanks to the ROS Master node, which provides naming and registration services to the rest of the nodes in the ROS system. Following with the previous example, the camera driver advertises that it wants to publish to the topic. Then, the other node communicates with the Master to subscribe to that same topic. Then, a peer to peer connection is established between these two nodes to send and receive the messages.

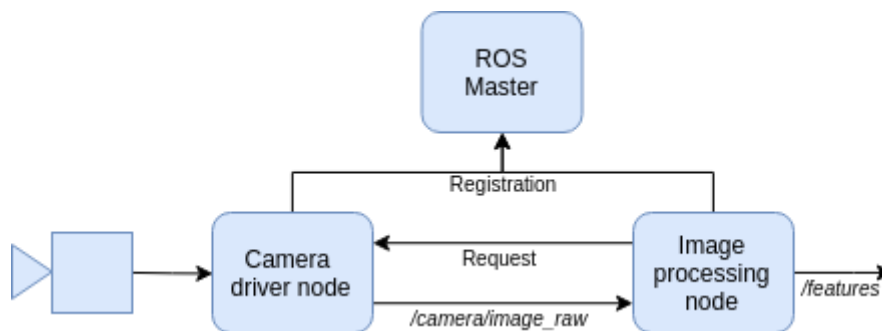


Figure 10. ROS structure example

This manner of working provides a lot of versatility, as the nodes can be physically running in different computers. This allows to split the computational charge between multiple computing units, providing they are connected through Ethernet or a wireless connection.

3.1.2. Rosbag: record and playback

Rosbag is a developing tool that allows the user to record all the messages that are sent during a certain duration. All the recorded messages are saved into a `.bag` file and can later be replayed, in the same chronological order that they were recorded.

This tool is very useful because the recorded data can be used to test different implementations and algorithms far away from the full setup, although the timing might not be the most precise because, as stated above, the messages are played back on the timestamp that they were recorded, not the timestamp they were sent.

Also, the `rqt_bag` tool can be used to play the messages with a user-friendly GUI that shows a time line with the messages, thumbnails for the image topics and also can be used to visualize raw data, along with a series of buttons to control the time.

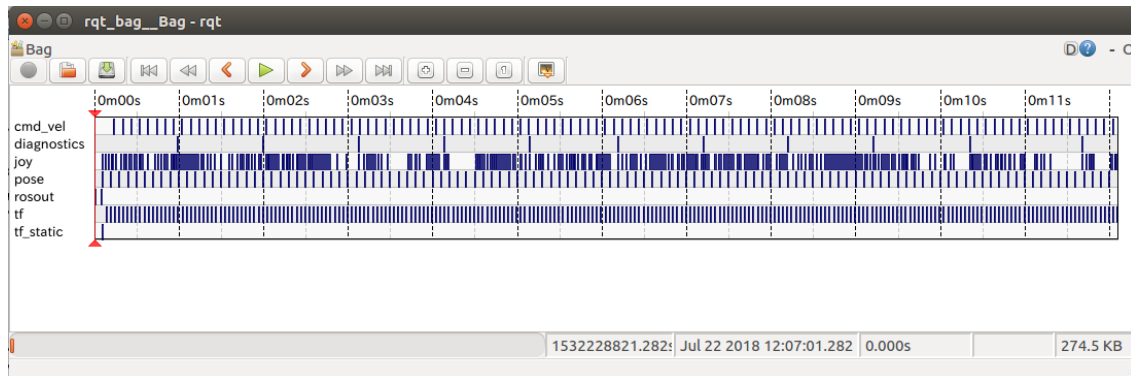


Figure 11. Timeline of rosbag shown on `rqt_bag`

3.1.3. Rviz: powerful visualization

Rviz is the ROS visualizer. It can visualize the most important ROS topics in real time, which can be especially useful when dealing with data types such as images or point clouds. Moreover, customized 3D markers can be added in order to visualize the computed trajectories of a trajectory planner or the detected objects in a point cloud, for example.

Rviz integrates a main screen that represents the 3D space and can display topics such as point clouds, positions, markers, etc. Moreover, if these topics have different systems of coordinates, Rviz can display them in the same system if the user provides a TF that relates each system of coordinates. This helps preserve the modularity of each node, while providing a fundamental tool for integration and visualization.

Also, image topics can be added in separate secondary displays.

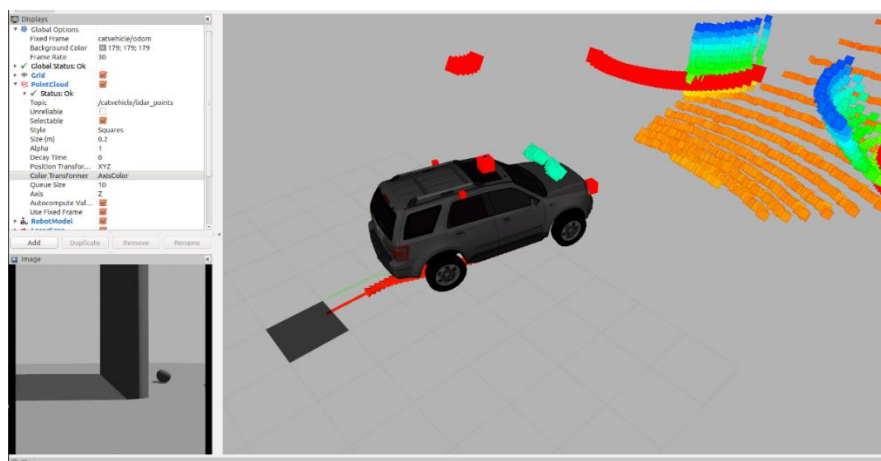


Figure 12. 3D model of a car and its environment acquired by sensors in `rviz`

3.1.4. Nodelets: zero-copy transport

The same characteristics that give ROS a lot of versatility cause an important drawback when transmitting large amounts of data. The fact that all the data is transmitted using TCP sockets can cause the network to saturate, even using only one computer. For small packets of data that don't represent a big amount of traffic, such as the torque and rpm that need to be applied to a motor, this doesn't present a big issue. However, when working with images and point clouds, major delays can occur and finally cause the application to crash.

There exist two different approaches to tackle this problem.

The first one is the Image Transport. In the case of images, this class can be used to compress the image stream in two ways, either compressing each individual image (using JPEG or PNG) or turning the images into a video stream with the Theora codec. The first causes less delay but achieves a worse compression ratio, whereas the second one provides better compression but a lot more delay.

The other option are Nodelets. These are a generalization of the typical ROS nodes. They are designed to provide a way to run multiple algorithms in the same process with zero-copy transport between them in the same computer. These nodelets need to be contained in one single process - the nodelet manager - that will run them in a shared memory environment, in order to communicate through boost shared pointers.

To do this, nodelets allow dynamic loading of classes into the same nodelet manager, while providing simple separate namespaces such that the nodelet acts like a separate node, despite being in the same process. Moreover, these are dynamically loaded at runtime by using the pluginlib library that can open other libraries containing exported classes at any point without the application having any prior awareness of the library or the header file containing the class definition.

This represents two clear advantages in comparison to the regular nodes: using zero-copy transport does not contribute to the saturation of the network and also there is no need to serialize, pack and send the data, eliminating the delay that these processes cause.

3.1.5. Message filters: synchronization between topics

Usually, when working with different sensors or many different topics, a synchronization point needs to exist. For example, a node that computes the disparity between two images from two different cameras will need to start when both images arrive, possibly with two slightly different timestamps.

The ROS Message Filters library provides a system of buffers that store a number of messages of a specified topic in a queue. Then, given a type of filter, the messages are forwarded or discarded.

The most used filters are the Synchronization Filters. These type of filters need a sync policy that determines how the messages need to be synchronized, which can be either exact or approximate. The ExactTime policy waits for messages that have exactly the same timestamp, down to the nanoseconds, while the ApproximateTime policy uses an iterative algorithm to synchronize the messages, assuming they can have different timestamps. This last policy can be used, for example, to synchronize topics with delays, jitters, different frequencies, etc.

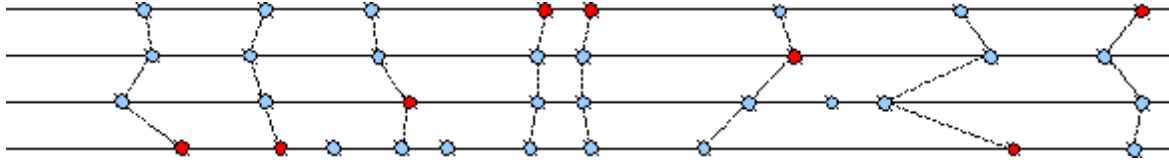


Figure 13. ApproximateTime sync with jitter [5]



Figure 14. ApproximateTime sync with different frequencies [5]

3.1.6. OpenCV: optimized computer vision

OpenCV is an open source computer vision and machine learning software library. It includes more than 2500 optimized algorithms, ranging from classical computer vision to state of the art machine learning algorithms. It is written in C/C++ and it is supported and used by some ROS libraries. All these characteristics make it a fundamental tool for the project, specially for the camera calibration and 3D reconstruction [6] algorithms that it provides.

3.1.7. PCL: cutting edge point cloud processing

The Point Cloud Library [7] is an open source point cloud processing library. It is the equivalent to OpenCV when talking about point clouds. Even though it is not as developed as OpenCV, it shows great potential and it is already being used in research and in the industry. Moreover, ROS uses PCL for most of the visualization that rviz provides, and also in many of its libraries.

4 DEVELOPMENT

4.1. PHYSICAL SETUP

4.1.1. Sensors used

The perception of the vehicle is built upon two main sensors, a LiDAR and a stereo camera. These sensors have some weaknesses and strengths and their combination offers a good trade-off.

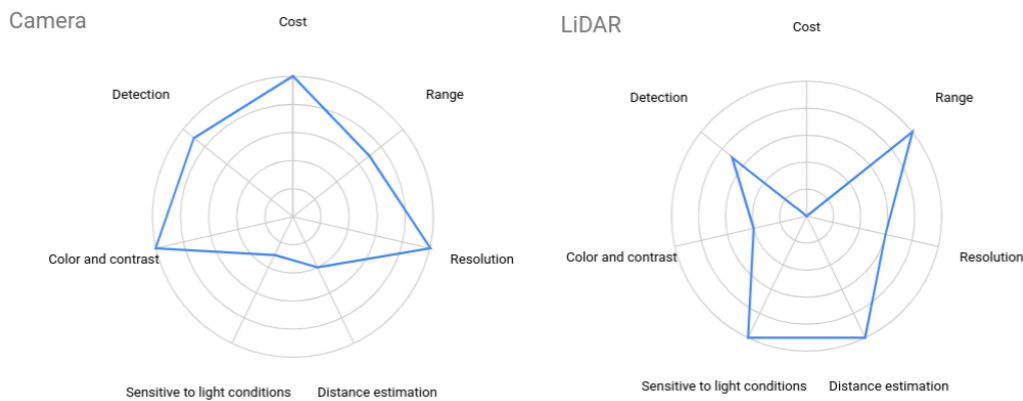


Figure 15. Trade-off between the characteristics of the used sensors

The main strengths to exploit of the camera sensors are their resolution, affordable cost, colour and contrast. Also, the strongest point is the already existing literature on image processing, detection and classification in comparison to the LiDAR point cloud processing, which is still not as developed. In contrast, LiDAR sensors have great distance estimation, low sensitivity to light conditions and range, making them very good and robust choices for the sensor layout of the autonomous vehicle.

The used LiDAR is a Velodyne VLP-32C. It is widely used due to its excellent specifications. It has a range of 200 meters and an excellent field of view of 360° horizontally and 40° vertically, which is the angle that the sensor is able to perceive. This is significantly useful in closed curves, in which the system is able to detect most of the cones in the curve to obtain a proper mapping and trajectory. The LiDAR allows spinning frequencies from 5 to 20 Hz. Also, the point cloud is divided in 32 layers that, in this case, correspond to the number of rotating lasers that scan the environment. The working frequency affects the resolution these layers provide, for example, at 10 Hz a point cloud has around 30.000 points and at 20Hz it can be ten times this size. This is a relevant constraint because the processing unit must be able to process all the data, which is transmitted via 100Mbps Ethernet using UDP packages.

The stereo vision system incorporates two independent colour cameras that are used as a stereo pair. The chosen cameras are two DFK33UX252 from *The Imaging Source*. They have a global shutter, meaning that the sensor is exposed to light all at once, avoiding the rolling shutter effect that appears in high speed applications. This model also allows the hardware trigger, meaning that the images taken from the camera are shot by using an external electrical signal, allowing a master device to control all the cameras to shoot at once. The resolution goes

from 640x480 to 2048x1536 pixels, and the framerate is limited by the available bandwidth of the USB 3.0 interface that the cameras use.



Figure 16. Velodyne VLP-32C (left) and 2x DFK33UX252 (right)

The lenses used are two 5MP low distortion board lenses with a format of 1/2 inches and a focal length of 3.4 mm. The parameters of these lenses are designed to acquire field of view of 96° vertically and 113° horizontally. This FOV, however, is then reduced because of the stereo configuration, depending on the baseline of the cameras and the resolution used. At a resolution of 640x480, for example, the vertical and horizontal FOVs are 76.5° and 89.2° respectively.

Finally the maximum range of the sensors, taking into account the size of the targets to detect and the stereo parameters of the developed setup is summed up by the following figure.

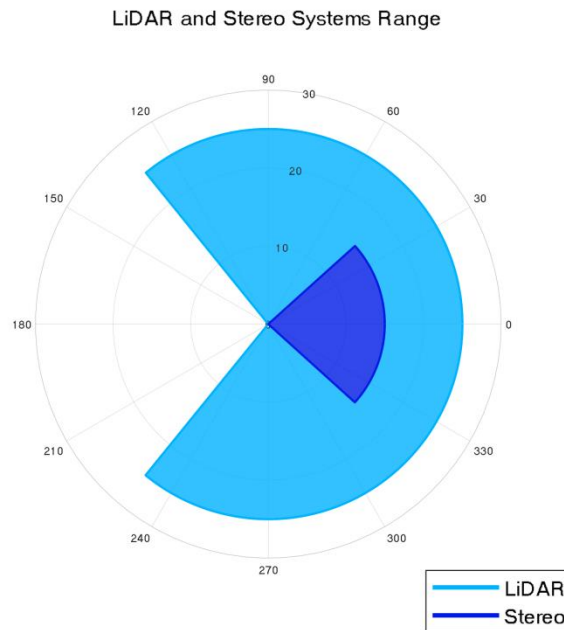


Figure 17. Range and FOV of the developed perception systems

4.1.2. Processing units

It is important that the processing units are powerful enough to process the data from the sensors and run all the algorithms involving the perception and control of the vehicle at a high computing speed. For some computer vision algorithms, such as neural networks, GPU processing is much faster than CPU. For that reason, a NVIDIA Jetson TX2 is used to carry out these calculations, as it has a 256 core graphical processing unit and 8GB of RAM memory. With that, it can run the inference of a neural network at around 30 fps (depending on the size and architecture).

For other perception algorithms, such as the LiDAR pipeline, as well as all the Estimation and Control algorithms, a rugged device with an Intel i7 processor and 16GB of RAM (Cincoze DX-1000) is used. These two computing units are interconnected via Ethernet through a switch and enclosed in a box that is resistant to dust and water.



Figure 18. NVIDIA Jetson TX2 (left) Cincoze DX-1000 (right)

4.1.3. Support and casing

The cameras should have a good perspective of the cones, that's why they are placed on the main hoop of the vehicle. In this part of the vehicle, the FSG rules highly regulate the positioning of sensors. All sensors have to be mounted within the rollover envelope, and in a maximum distance of 500 mm above the ground and less than 700 mm forward of the front of the front tires, as can be seen in Figure 19.

The support for the cameras consists of an aluminium plate that contributes to the cooling of the cameras, that is supported by two 3D printed braces that are held onto the main hoop. Also, two 3D printed covers for the cameras with a methacrylate front part that allows proper visibility.

Regarding the LiDAR support, it is mounted on the front wing with an aluminium support, fabricated with laser cut and welded together.

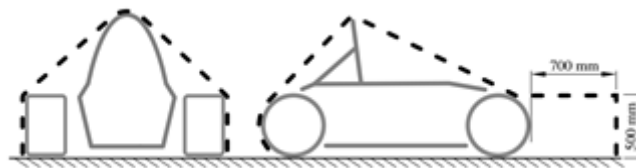


Figure 19. Envelope to mount sensors



Figure 20. Stereo camera and LiDAR supports

All The processing units are enclosed in a box that is mounted behind the main hoop. These are interconnected by an Ethernet switch. The Ethernet connection between the two computers is mandatory for the ROS distributed architecture, so that both can share topics and services. The switch is useful because it can either be connected directly to a third PC for configuration or can be plugged into a wireless router to do it at longer distances. The connection diagram can be seen in Figure 21.

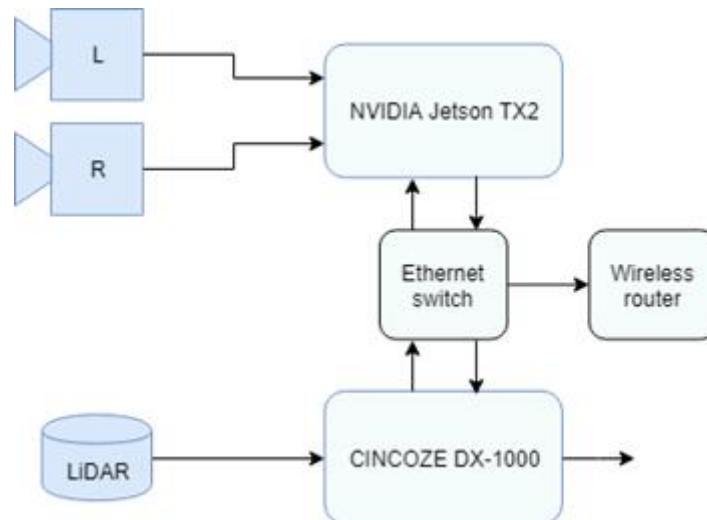


Figure 21. Connection diagram inside the PU box

4.2. PERCEPTION SYSTEMS

The perception of the vehicle consists on the following diagram. This system processes the data from the two types of sensors in three different streams, the LiDAR pipeline, the Stereo pipeline and the Sensor Fusion pipeline. This is done in this way due the requirement of robustness, as different approaches can tackle the weaknesses of the others.

In the best case scenario, the Sensor fusion pipeline can be the most accurate of the three, but in the case of a sensor failure, harsh weather conditions or other complicated scenarios, it may be better to run different approaches.

Moreover, these pipelines are designed to be able to be run in parallel, so that the detections provided can be compared between each other.

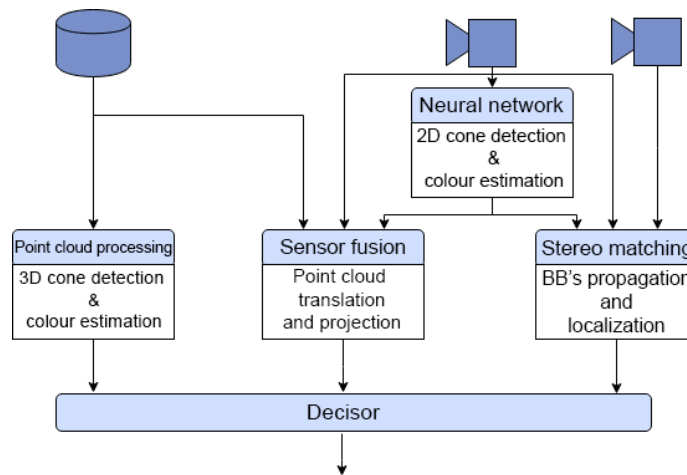


Figure 22. Perception block diagram

4.3. STEREO PIPELINE

4.3.1. Introduction

In this section, an in-depth exploration on the Stereo pipeline will be made. All the nodelets, their connections and the internal steps will be explained, in addition to their purpose and why many of the design choices were made. But first, a brief overview of the system will be done.

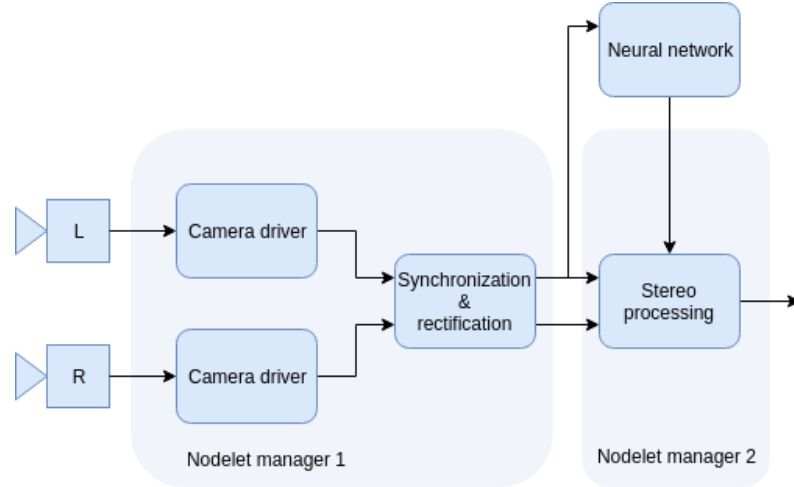


Figure 23. Stereo pipeline diagram

To start with, both cameras need to be calibrated. Knowing well their intrinsic and extrinsic parameters is a fundamental step for many of the following algorithms. That's why there is a first offline phase in which these parameters are carefully obtained.

To begin with the online phase, a self-developed ROS camera driver is used to capture the images and introduce them in the ROS environment. Each camera has its own instance of this driver and they publish images asynchronously. To obtain synchronous image pairs, an ApproximateTime synchronizer is used. This program also cancels out the distortion caused by the lens and rectifies the images, meaning that they are projected onto the same plane.

Then, a convolutional neural network is applied on the rectified images from the left camera, resulting in the detection of cones and its colour. Then, the intrinsic and extrinsic parameters of the stereo camera, as well as the previous knowledge about the cones' size, are used in order to obtain an estimated detection on the right camera image.

As the neural network is the bottleneck of this pipeline, this approach is less time consuming than running the neural network cone detection on both left and right camera images. Moreover, the bounding box propagation to the right image provides a coarse estimation of the 3D position of the cones, which is later used as a backup if the stereo distance calculation could not be obtained.

Once the same detection on the two images is obtained, a local feature matching algorithm is applied, along with a triangulation. This final step estimates the final position of the cones. In case no successful matches were to be found, the previous coarse position estimation is used.

4.3.2. Camera calibration

The camera calibration stage is fundamental for all the stereo computations, and also to correct the lens distortion and to allow a better detection for the neural network. This stage is performed offline, and the information obtained is stored to be used in the online phase.

The objective is to characterise the camera using a pinhole model. This model uses the focal length f , the spatial sampling frequencies $[s_x, s_y]$ and the principal point of the camera (the image centre) $[c_x, c_y]$. These parameters in the configuration of the intrinsic matrix, usually referred as K , are an approximation of how the points in 3D space are projected onto the

image. So, being $[x, y]$ the pixel coordinates, λ a scaling parameter and $[X, Y, Z]$ the 3D world coordinates, the following formula is obtained.

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f s_x & 0 & c_x \\ 0 & f s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Equation 1. Intrinsic camera matrix formula

More complexity can be added to the model, by using the distortion coefficients k_c that model the distortion that is added by the lens. This model takes into account the two most common types of circular distortion, barrel or pincushion.

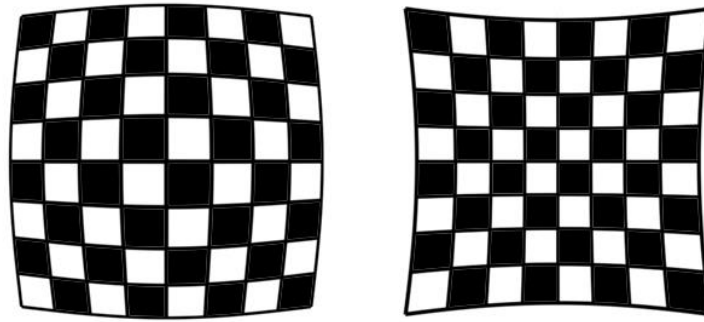


Figure 24. Barrel distortion (left) and pincushion distortion (right)

These kinds of distortion can be distinguished by using a checkerboard pattern as shown in the image above. Also, the alternating colour of the squares is used to detect corners and, therefore, they can be used to estimate the intrinsic parameters of the camera, as their real size is known.

Once the intrinsic parameters of both cameras have been obtained, some more parameters have to be computed for the stereo camera setup. These are the relationships between their optical centres and their relative rotation, modelled by the T and R vectors.

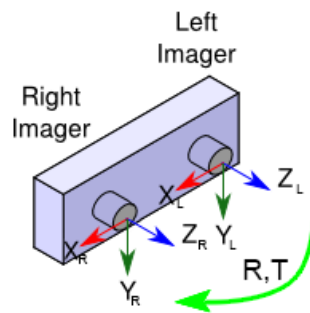


Figure 25. Extrinsic transform between left and right imagers

For the calibration process, the MATLAB stereo calibration tool is used. Based on around 20 pairs of synchronized images of a checkerboard pattern, this tool computes the intrinsic and extrinsic parameters of the stereo camera that optimize the reprojection error, which is the error between the actual detected corner points of the checkerboard pattern and the ones that

are projected again after the 3D estimation, by using the parameters and the Equation 1. The reprojection error is depicted in Figure 26.

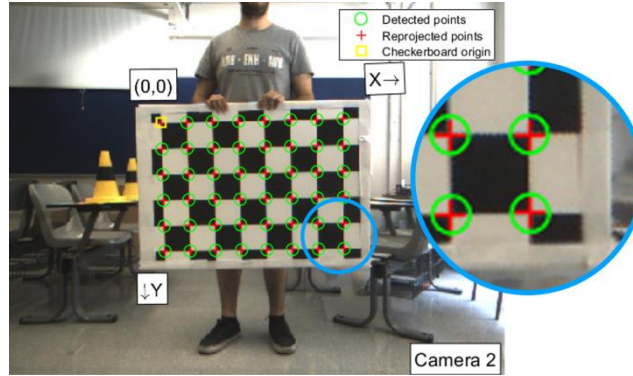


Figure 26. Reprojection error as the distance between the detected and reprojected points

Finally, these parameters are now used for the image rectification. The goal is to apply a projective transform to both images so that they are onto the same plane. This means that the epipolar lines are horizontal, and the projections of any point in space share the same y-coordinate on both images. This fact is widely exploit among many stereo processing algorithms to reduce searching areas, make calculations simpler, etc.

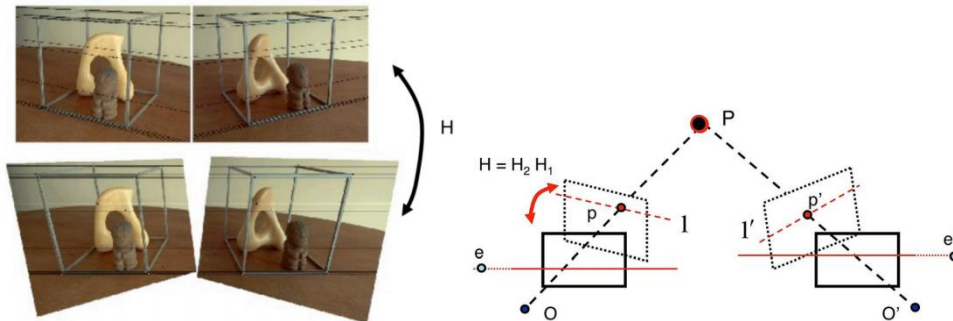


Figure 27. Image rectification. Epipolar lines are horizontal on the rectified plane [8]

4.3.3. Image acquisition and trigger

The first step of the pipeline is the image acquisition. The aim of this stage is to get the images from the camera and then introduce them into the ROS framework. Also, an important requirement for the stereo image processing is that the cameras need to be synchronized. In consonance with these requirements, a self-made driver was developed in order to ensure their fulfilment, as the already available tools didn't.

The driver uses the well-known GStreamer library [9] to obtain the images and the open source *tiscamera* SDK [10], developed by the manufacturer of the cameras, to communicate with their devices and set their properties or adjust their parameters.

First, the GStreamer pipeline is started, along with the ROS publishers to provide the images to the other nodelets. Then, a time rectification is applied between the ROS and the GStreamer timestamps. For the synchronization, it is a better option to use the GStreamer timestamps as

they come from a lower level and are less subject to possible delays, but these timestamps are relative, as they start when the GStreamer pipeline is opened. After the time rectification, the images have the more precise GStreamer timestamp but in relation to the Unix Time [11], that is used by ROS.

Then, for the triggering of the cameras there are two possible approaches: a hardware trigger that makes the two cameras shoot at once or a more passive approach that lets the cameras send their frames independently at a constant rate.

The first approach is better in terms of synchronization because it is guaranteed that the two frames that are closest in timestamp are shot at the same moment, whereas the second one is subject to an error in time of half the period of acquisition in the worst case, which can lead to significant effects depending on the speed of the vehicle. For example, at 15 m/s and 30 fps, the error is $e_p = \frac{1}{2 \cdot 30} s * 15 m/s = 0.25 m$ which is unacceptable, but can work in steady conditions for testing.

The hardware trigger normally requires an external source to give a signal to both cameras at the same time at a determined rate. However, in this case, both cameras have GPIO pins that can be configured in a master-slave configuration. One camera will trigger at a constant frame rate, for example at 15fps (master), while the other (slave) will trigger when it receives the trigger signal from the master. In this way, the delay between both cameras' triggers is minimized, allowing a better synchronization.



Figure 28. Free-running cameras timestamps

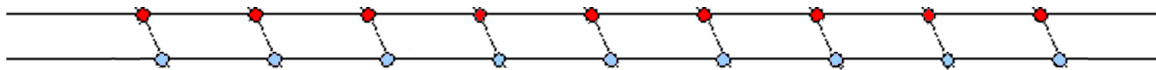


Figure 29. Hardware triggered cameras timestamps

4.3.4. Synchronization and rectification

Once both camera drivers are publishing the images, there is a need for synchronism as the following nodelets require a pair of images as input topic and they may have slightly different timestamps. For this purpose, the synchronization and rectification nodelet uses an ApproximateTime synchronizer. Then, the timestamp of both images is set to the mean value between them. This is done because all the following nodelets will use an ExactTime synchronizer. The use of ExactTime instead of ApproximateTime is preferred because the exact produces less delay than the approximate, as it doesn't have to use an iterative algorithm to find the image pairs. Moreover, setting the same timestamp to both images will grant that the same pair of images is chosen at the input of the following nodelets.

The next step is the rectification and lens correction. As these operations involve many matrix products, they are computationally expensive. A common practice is to use precomputed look up tables. These tables are transformations that map the pixels in the original image and the rectified and undistorted image. The use of this method substitutes floating point operations for individual pixel mappings that make the computations way faster. Moreover, an implementation of this inverse mapping is available in the OpenCV 3D Geometry and Stereo Vision chapter [12].

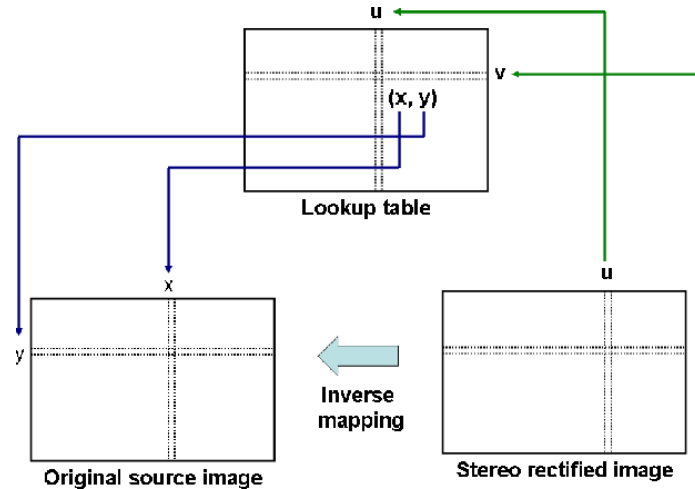


Figure 30. Inverse mapping to obtain the lookup table [13]

Once the images have been paired up and then rectified and undistorted, they are published using the Image Transport library. In this step, the rectification of the images can be checked by tracing epipolar lines, as it can be seen in Figure 31.



Figure 31. Epipolar lines in a pair of rectified images

4.3.5. CNN Cone detection

The next step in the pipeline is the cone detection that uses a convolutional neural network developed by *R. Aylagas*. This network is trained to detect four different classes which correspond to the four types of cones that need to be identified and localised, in Figure 32. This method for the detection was chosen according to the conclusions of the work developed in the PAE final project, in which a first simple contour-based cone detection approach was developed and tested. The need for a detection that is reliable, fast and resilient to weather conditions supposed that a CNN-based detection needed to be used.





			
big orange cone two white stripes	small orange cone single white stripe	small yellow cone single black stripe	small blue cone single white stripe
WEMAS 307.610500.00.00	WEMAS 400.000013.00.00	WEMAS 400.000013.01.10	WEMAS 400.000043.00.00
285 mm × 285 mm × 505 mm 1.05 kg	228 mm × 228 mm × 325 mm 0.45 kg		

Figure 32. Colour and physical characteristics of the cones

This network was trained using an open source dataset [14] of around 18000 images of cones that was developed by FSD teams. More than 800 of these images were shot by the Driverless UPC team, using the cameras that will be in the car and then they were manually labelled.



Figure 33. First training of the neural network

The implementation of this neural network is done in Python, which provides a very easy way to interact with both the GPU and the CPU of the Jetson TX2. This allows the use of a multi-threaded approach to the neural network, doing a very interesting use of the available resources to obtain an inference rate of around 30 fps. The downside of this detection is that Nodelets aren't available in ROS python. In the absence of Nodelets, for the communication to this node, the Image Transport library (detailed in section 3.1.4) is used. The left images, which are the ones that this node uses for the detection, are transmitted through the Jetson TX2 network interface, but are compressed to ensure that no network saturation is produced.

4.3.6. Stereo processing

The final algorithm of the pipeline is the stereo processing itself. It takes as input a synchronized pair of images and an array structure that contains the detections made by the neural network.

The first step is the bounding box propagation. This means using the previous knowledge about the cones' size to obtain an estimate of the detections on the right camera image.

This part of the system works very similarly on how the Instagram face filters do, in which some keypoints are extracted from the face and knowing their relative position, these

keypoints can be extrapolated to a 3D model and estimate the tridimensional position of the face in relation to the camera.

For example, the most common keypoints are the eyes, nose, lips and chin. Once these are found, the next step is to match these 2D positions in pixels to the average 3D position of these features in real people.

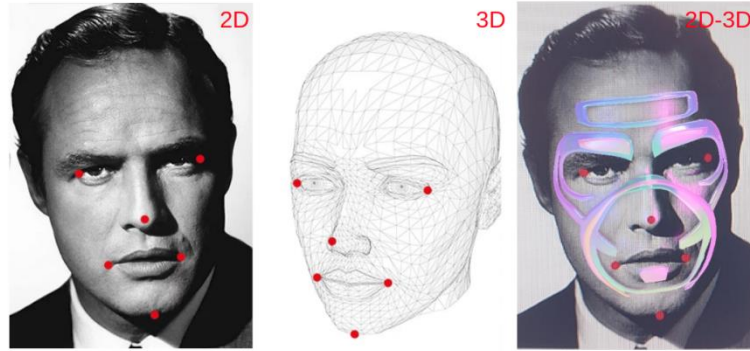


Figure 34. 2D-3D correspondences of a face filter to insert an augmented 3D object

This step is normally done by neural networks that learn features about the target and then reproduce them in new instances of that same target. At the start of this project this was intended to be done in this way but due to the occupation created by the CNN cone detection on the Jetson TX2 GPU, this approach was discarded and a much simpler yet effective approach was found.

Instead of regressing some keypoints from the cones, the bounding box corners provided by the detection are used. The received bounding boxes from the neural network are assumed to have approximately the horizontal and vertical dimensions of the cones. This allows the 2D-3D EPnP algorithm [15] to estimate the position of a cone by matching the corners of the bounding box to its 3D model with its real dimensions (228 mm x 325 mm for the small ones and 285 mm x 505 mm for the big ones).

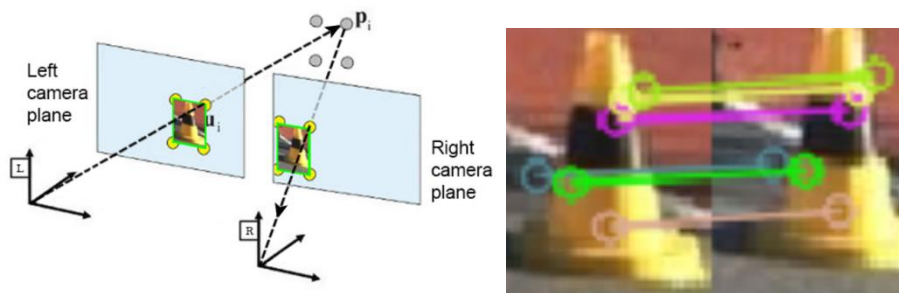


Figure 35. Bounding box propagation scheme (left) and feature matching (right)

The algorithm used, the Efficient Perspective n Point, is a very fast solution to the PnP problem, providing a rough approximation for the position. However, the goal here is not to be accurate, but to quickly extend the neural network detection from one image to the other. This is now done by projecting the 3D bounding box approximation to the right camera, using the stereo camera parameters. With this propagation, a local feature extraction and matching can be performed. That is, using feature extraction on both bounding boxes and a KNN matching algorithm to establish pairs of points. These pairs of points are restricted to be on a range of ± 5 px vertically, as they are assumed to trace epipolar lines, due to the rectification of the

images. Finally, those points are triangulated and their centroid is estimated to provide the final and precise estimation.

In some cases, it can occur that no good matches are found on the image. This is solved by using the first rough estimate of the position that was used to propagate the bounding box. This step provides robustness to the algorithm, that is be able to function on all the cases that no matches are found but more importantly it can operate with only one camera, in the event that one of the available cameras were to fail.

4.4. SENSOR FUSION

4.4.1. Introduction

The aim of the sensor fusion system is to combine information of both sensors, cameras and LiDAR, to obtain a more reliable estimation of the cones on the track. The CNN cone detection of the camera pipeline has a much better accuracy in comparison to the point cloud processing algorithms that the LiDAR pipeline uses. However, the positions obtained using the stereo camera are susceptible to the sub-pixel calibration errors, which can have an effect of meters at long distances. These characteristics of the systems can be fused together to increase the results.

The need to combine information from both sensors is also important because the information that each independent pipeline provides is relative to its own coordinates system and, therefore, a common coordinates system can be used to express the positions of all the detected cones. This system can then be used by the Estimation and Control department to obtain an approximate map of the environment.

The registration between the sensors can be expressed as the relative transformation between their coordinates axes of the type: $\xi_{cl} = (t_x, t_y, t_z, \phi, \theta, \psi)$, which are the translations along each Cartesian axis and the three angles of rotation. This is a difficult to obtain transformation, as the optical centre of the stereo camera is not a physical point on the device, neither are their axis. This coordinate system emerges from the camera model, which is something abstract. Similarly, the LiDAR approximation of the optical centre and its axes provided by the manufacturer is not precise enough to obtain the transformation.

To solve this problem, both devices need to be set in a fixed relative position. Then, similarly on how the camera calibration works, a calibration pattern can be used to obtain a mapping between the points detected from one sensor to the other. However, for this specific calibration problem there is no “standard” pattern to use. There exist many different ways to obtain this registration such as checkerboard or circle patterns. However, the final decision was to use the pattern proposed by *C. Guindel et al* [16], which is a flat board with four circular holes.

Once the calibration is solved, the obtained registration can be used to project the LiDAR point cloud onto the left camera plane. Then, the points that fall into a bounding box in the image are treated like a cluster, not only with a very reliable detection provided by the neural network, but also knowing its colour and maintaining the good position estimates provided by the LiDAR sensor.

4.4.2. Calibration

As stated above, the calibration stage uses the set-up proposed by *C. Guindel et al* [16]. To calibrate a stereo camera and a LiDAR sensor. These two sensors need to generate a point cloud so that the detected features from the calibration target can be identified on both sensors and obtain a proper transformation.

The process starts with the point cloud generation. For the LiDAR this step is automatic, as the output from the sensor itself is already a set of points with XYZ coordinates. For the stereo camera, however, this requires some previous steps. Like in all other stereo processing algorithms, the intrinsic and extrinsic parameters of the cameras need to be known. Then, the image pairs are rectified and processed by using the semi-global matching algorithm (SGM) [17]. This approach is similar to the block matching algorithm, but instead of naively looking for matches, this method imposes a pixel-wise cost function that helps to find the best match, in combination with a continuity constraint that penalizes abrupt changes in the disparity.

Once both point clouds have been obtained, the next step is to filter them. Both point clouds are filtered by range according to the approximate distance at which the pattern is placed to eliminate the points that don't belong to the pattern. Then, a plane RANSAC is applied to further discard the outliers and finally, RANSAC is used again to find the final points that may be part of a circle and obtain its centre. Particularly, the stereo camera also uses the left image gradient to obtain a better estimation of the borders of the pattern. This whole process can be seen in Figure 36.

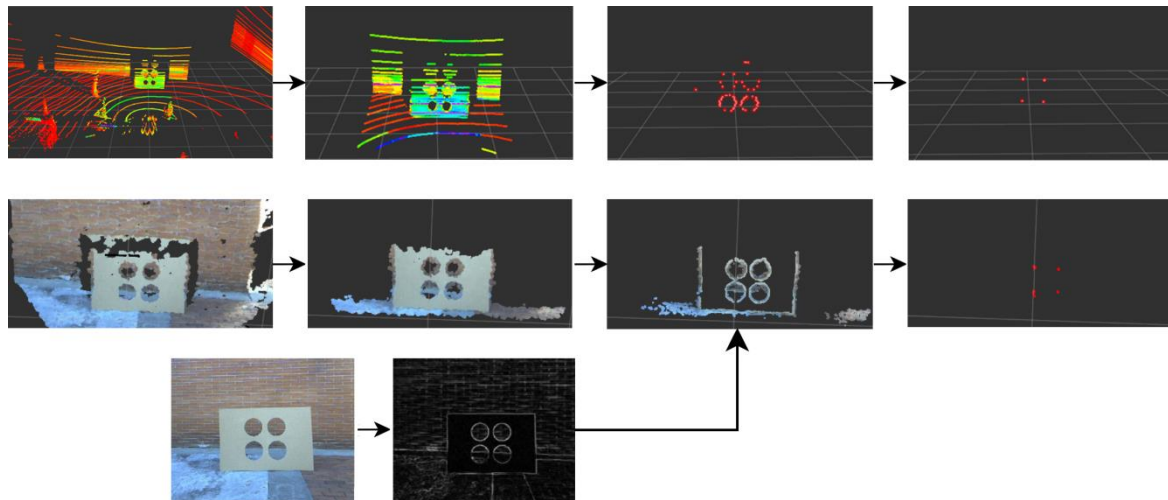


Figure 36. Processing of the LiDAR and Stereo point clouds

After the processing, the obtained centroids are stored in two separate cumulative point clouds. When there are enough instances, the translation that minimises the Euclidean distance between the respective centroids is obtained by computing the least-squares solution of the overdetermined system of 12 equations provided by the registration of the four reference points. Using this purely translational model as a first step is convenient because it gives a first approximation of the final transformation function. After that, the final transformation is obtained by using a rotational and translational algorithm known as Iterative Closest Points [18].

4.4.3. ROS implementation and point cloud projection

The ROS implementation for this part of the pipeline is multi-platform because the convolutional neural network is running on the NVIDIA Jetson TX2 and all the other algorithms are running on the Cincoze. This requires that a minimum amount of data is transmitted. This is solved by only transmitting the detection (the bounding box corners and the detected class) and the camera parameters to the Cincoze, and, this way, the link between these two processing units is not saturated. Moreover, both devices need to have the exact same time, as the received ROS topics need to have timestamps as accurate as possible to get synchronized by a Message Filter. For that, NTP packets are used. The diagram for the implementation in ROS for this pipeline can be seen in Figure 37.

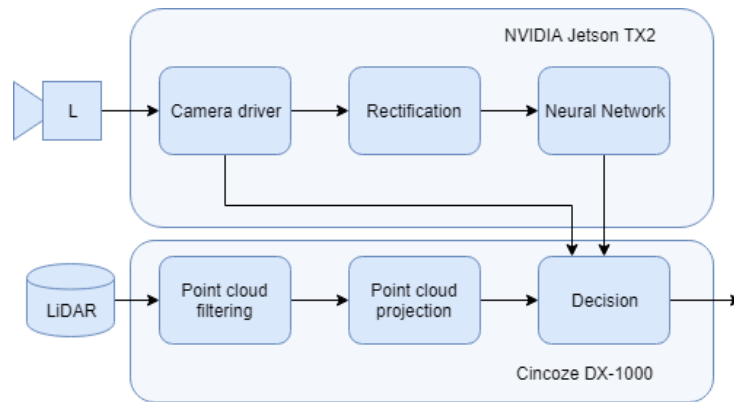


Figure 37. Sensor fusion ROS diagram

Once the extrinsic transform between the sensors has been obtained, the sensor fusion algorithm can project the filtered point cloud to the image plane and use the bounding boxes detected by the CNN to determine that a cluster is indeed a cone and extract its colour. In Figure 38, the CNN detections (bottom left), the projected point cloud (top left) and the full point cloud with the final detections (right) can be seen. In this case, a slight calibration error can be appreciated on the top left image. However, the performance of the algorithm is not compromised, because even if only a few points of a cone are projected inside the bounding box, as there are a lot more points in comparison to the stereo camera and their position estimation given by the LiDAR sensor is so precise, the final position for that cone is determined correctly. This run-time part of the sensor fusion system has been developed by A. Roche.

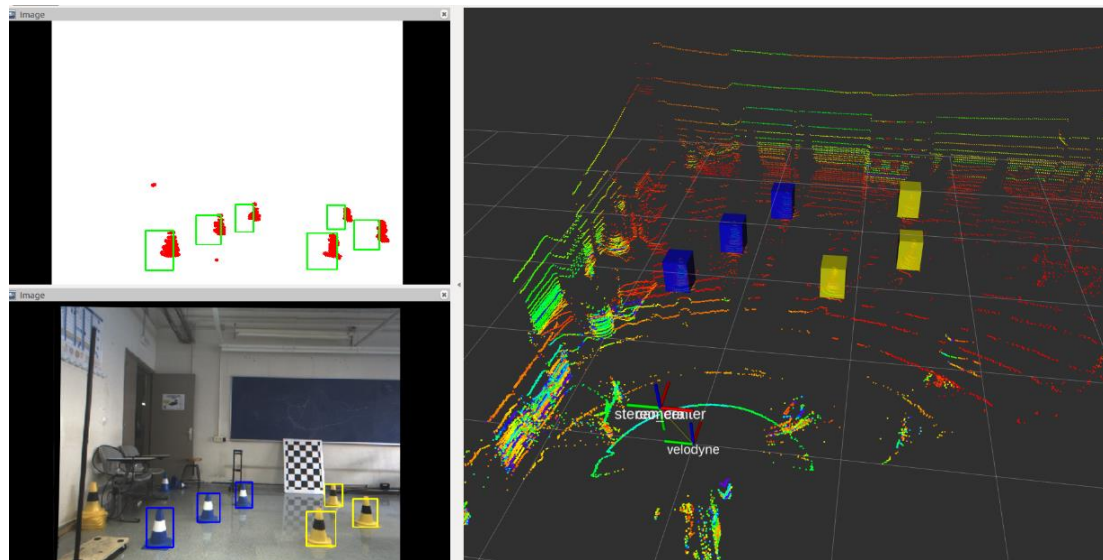


Figure 38. Sensor fusion: project the point cloud to the image plane and detection using the CNN

5 RESULTS

5.1. STEREO PIPELINE RESULTS

The results of the stereo pipeline will be presented in this section. To start, the root mean squared error in respect to a ground truth of cones will be compared to the theoretical RMSE based on the parameters of the stereo setup [19].

This test is a setup of cones closely positioned to the camera, ranging from 2 to 5 meters. After running the stereo algorithm for around 500 iterations, the results are the plotted in Figure 39.

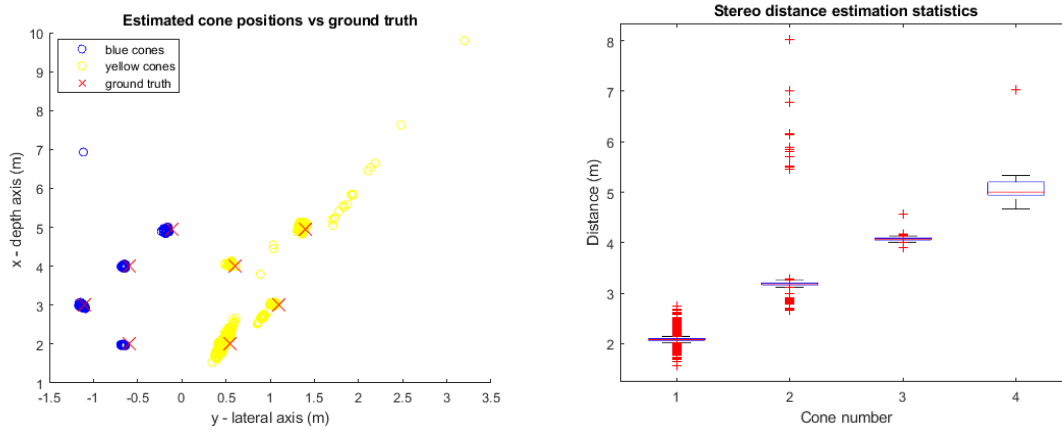


Figure 39. Estimated cone positions in comparison to the ground truth (left) and estimated distance boxplot (right)

The average error for these cones is rather inconsistent. While the blue cones on the left and the last two yellows show little to no error, the first two yellow cones show a lot of variance in their position estimate. Moreover, this variance can be seen mainly in the direction of the projection lines (from the optical centre to the position estimate). This is happening because of erroneous matches inside the bounding box, which could be mitigated by the use of a median filter for all the matches found. Also, this errors could be caused because the lenses' distortion grows at the borders of the images and the model for this distortion does not provide a precise enough calibration.

Using all the obtained data, the overall root mean squared error is 5,88 cm. However, if the two first yellow cones are considered to be outliers, the error would be of around 4,48 cm. This means that the accuracy of the position estimation increases significantly when no erroneous matches or calibration errors are found, and that these effects are in sight for future improvements.

The boxplot on the right shows that there is a lot of variability on the distance estimation, due to the problems mentioned in the paragraphs above. However provided that the mean value converges to the real position, at 15 frames per second there are enough position estimates so that the data from the same cone in different frames can be associated.

The theoretical error curve for the stereo camera setup can be seen in Figure 40, along with the results of this test. With a depth of 5m, the theoretical RMSE is around 12cm. The approach presented in this work for the stereo camera based position estimation shows less error than the expected. This could be because of the preliminary knowledge of the cones that is used

significantly improves the position estimates in comparison to other approaches such as the ones that are based on block matching.

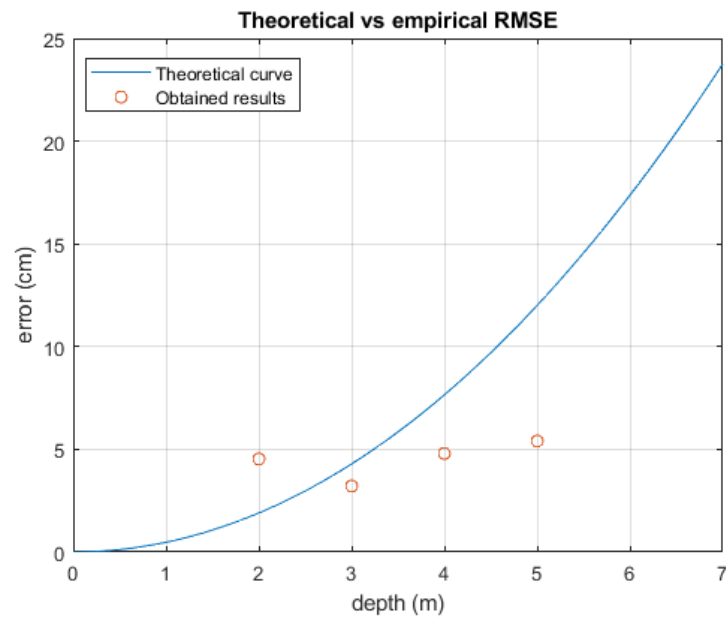


Figure 40. Theoretical vs empirical RMSE

Regarding the real-time requirement, the stereo system runs at 10-15 fps, maintaining a delay of around 70ms on average.

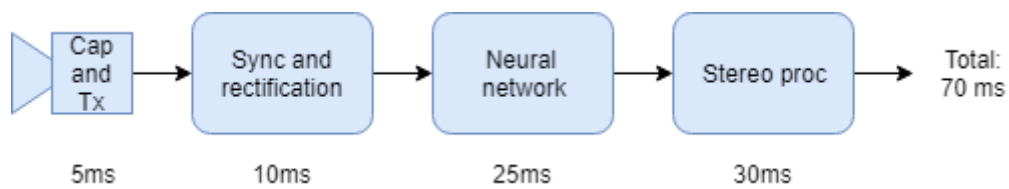


Figure 41. Timings of the stereo pipeline

5.2. SENSOR FUSION PIPELINE RESULTS

The sensor fusion pipeline will be tested in several ways. In the first place comparing the two registered point cloud, then by projecting the point cloud onto the image plane and in the final place evaluating the performance of the full sensor fusion algorithm in conjunction with the E&C systems.

To start, a coarse way to evaluate the results is to superimpose both point clouds and compare the different elements that are in the scene, such as the calibration pattern, in Figure 42.

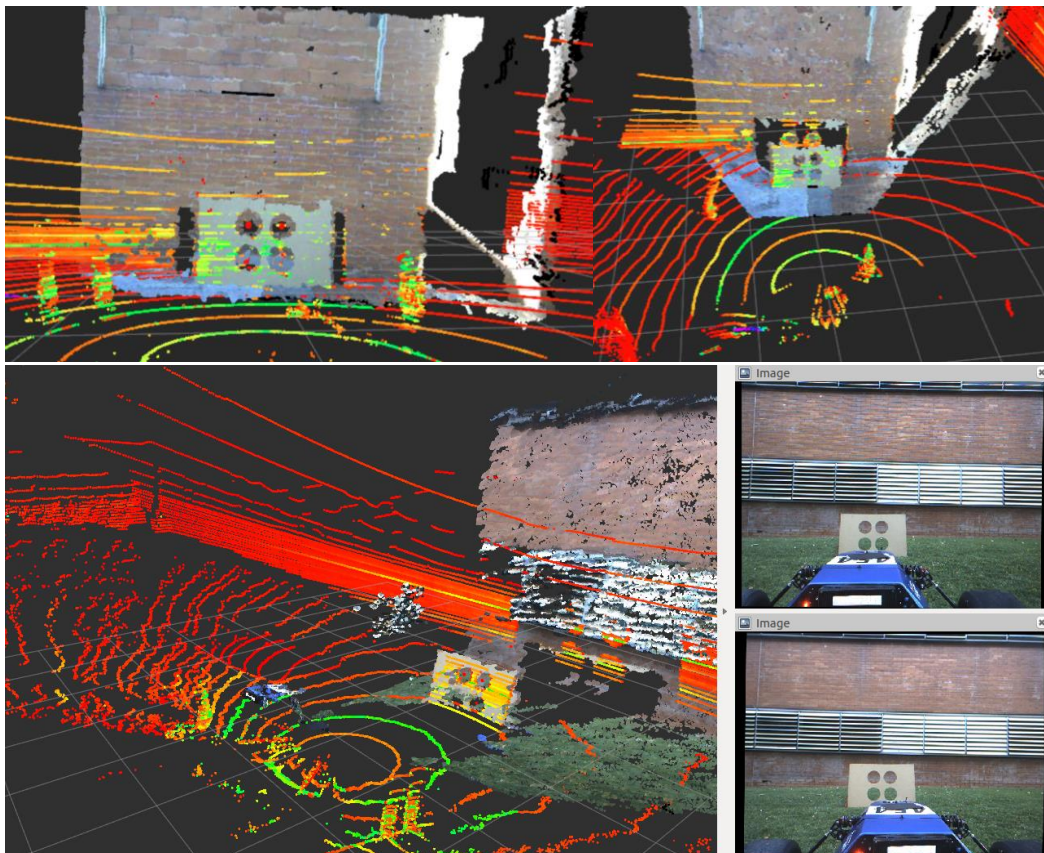


Figure 42. Two different setups of Stereo and LiDAR point clouds superimposed

Once the calibration stage has been performed, the point cloud projection can be tested, and the quality of the registration can also be evaluated, as in Figure 38, where the projected point clouds that represented cones were slightly out of position.

Also, the same test with the same ground truth as for the stereo system was performed. With eight cones at a known distance, the average RMSE is 3,68cm. This system slightly outperforms the stereo system when talking about average accuracy and furthermore, in terms of variance of the estimate it is way better as it can be seen in the boxplot on the right, in Figure 43.

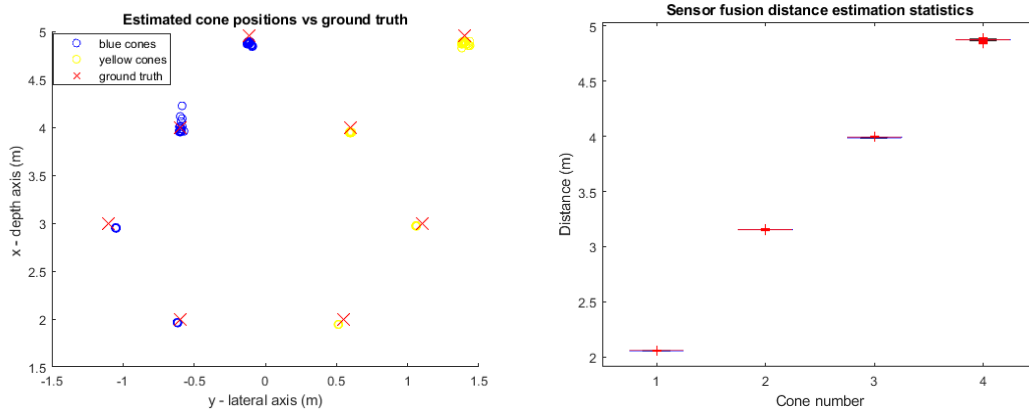


Figure 43. Estimated cone positions vs ground truth (left). Sensor fusion distance estimation statistics (right)

In relation to the speed of this algorithm, while it maintains its time of execution under 100ms, it is limited by the frame rate of the LiDAR sensor which, in this case, is of 10Hz for a full frame.

Finally, in Figure 44, a real test in which the control algorithms computed the track limits, trajectory planning, etc. All this with real data acquired by using the sensor fusion pipeline.

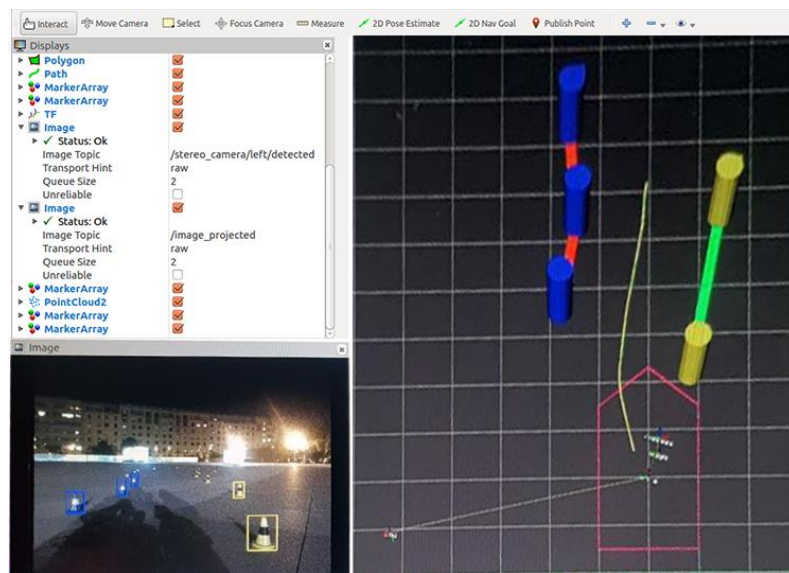


Figure 44. Track limits and path planning using the sensor fusion

6 BUDGET

All the software used in this project is under the BSD open source license. An exception is the MATLAB software and its calibration tool.

The hours for the involved engineering students are around 27h/week during 24 weeks, with a wage of 10€/h.

For the TFG tutor, 3h of weekly meetings during 24 weeks with a wage of 30€/h.

Plus all the materials and devices needed, the total budget equals a sum of 29.936,31 €. However, this is a university project and many companies want to be part of it by providing material in exchange for a sponsorship. The team has achieved more than 17.000,00 € in discounts, leaving a final cost of 12.224,50 €.

Concept	Qty	Cost	Real cost	Sponsor discount	Final cost
The Imaging Source DFK33UX252 Industrial colour camera + board lens + USB3 cable	2	1.200,00 €	2.400,00 €	100%	0,00 €
Velodyne VLP-32C	1	13.000,00 €	13.000,00 €	100%	0,00 €
NVIDIA Jetson TX2	1	350,00 €	350,00 €	33%	234,50 €
Cincoze DX-1000	1	5.000,00 €	5.000,00 €	33%	3.350,00 €
D-Link Ethernet switch	1	46,31 €	46,31 €	100%	0,00 €
MATLAB license / year	1	500,00 €	500,00 €	100%	0,00 €
Junior engineer (648h * 10€/h)	648	10,00 €	6.480,00 €	0%	6.480,00 €
TFG tutor (72h * 30€/h)	72	30,00 €	2.160,00 €	0%	2.160,00 €
TOTAL			29.936,31 €		12.224,50 €

Table 2. Budget

7 CONCLUSIONS AND FUTURE DEVELOPMENT:

To sum up, two subsystems for the perception of the Driverless UPC Formula Student car “Xaloc” have been developed in this project. The overall perception system is designed to fulfil the requirements of robustness and reliability, having three different ways to simultaneously provide accurate cone position estimates, while minimizing delays.

In retrospective, the systems that were developed in the context of this project, have been an excellent way to explore many aspects of the computer vision field: some classical methods were learnt, such as the pinhole model camera calibration and the keypoint extraction and matching, but also many state of the art techniques and algorithms were used, like the extrinsic Stereo-LiDAR calibration, the sensor fusion itself, etc.

In addition, the choice of self-developing a stereo camera instead of opting for a commercial one gives an added value to this system, because a deeper insight on how these kind of sensors work has been achieved, besides giving more versatility to the system. Also, the fact of cleverly using the prior knowledge of the cones’ size to obtain a rough position estimation is a very notable improvement. In addition to avoiding to run the neural network on both left and right images and thus optimizing the use of the resources, this method provides robustness to the algorithm in the case of a possible sensor failure. The results shown in section 5 prove that the performance of the stereo camera is adequate in terms of accuracy and speed. However, the system could be improved in many ways as discussed in the results section. For example, finding ways to reduce the variance of the position estimates, or increasing the range.

In relation to the Sensor Fusion pipeline, the implemented extrinsic calibration system between Stereo and LiDAR reliably provides the transformation function between them both. However, a future implementation of a calibration that doesn’t require a stereo point cloud could be considered, as the accuracy of such point clouds might not be as accurate as other methods of obtaining the distance. The provided registration of sensors has proven to be resilient to vibrations, and the slight miscalibrations that can occur in practise while operating the vehicle are easily corrected manually. When talking about performance, the overall Sensor Fusion pipeline outperforms the Stereo in terms of accuracy, but it is a bit slower, being able to run at a maximum of 10Hz.

In this kinds of projects where there is a need to integrate many different algorithms, there needs to be a way to put them together in the most efficient way possible. In this case, ROS provided the best way of doing that, thanks to its distributed architecture, standardized message types, powerful visualization tools, etc. The use this framework resulted in a very fast and efficient integration between different algorithms in the same department and even when integrating with the rest of the team. The learning curve of the Robot Operating System might be steep at first, but it is definitely worth the effort.

While this project finishes here, the Driverless UPC team will keep going on until the season ends in August 2019, after the Formula Student competitions in which the vehicle will be put to its ultimate test. The last objective of the project presented here is to grant the continuity of the Driverless UPC team, accumulating all the knowledge acquired, all the challenges that have been overcome and also all the mistakes that were made through this first season, to always keep in mind that *“It’s not only about going faster, it’s about getting smarter”*.

Bibliography:

- [1] "FSG," [Online]. Available: www.formulastudent.de.
- [2] A. B. M. P. C. E. M. G. R. S. A. G. M. P. M. B. I. S. R. D. R. S. Nikhil Bharadwaj Gosala, "Redundant Perception and State Estimation for Reliable Autonomous Racing," *International Conference on Robotics and Automation*, 2019.
- [3] A. F. Joseph Redmon, "YOLO9000: Better, Faster, Stronger," *CoRR*, 2016.
- [4] M. & C. K. & P. G. B. & F. J. & F. T. & L. J. & W. R. & Y. N. A. Quigley, "ROS: an open-source Robot Operating System," *ICRA Workshop on Open Source Software*, 2009.
- [5] "Approximate time sync - ROS," [Online]. Available: http://wiki.ros.org/message_filters/ApproximateTime.
- [6] "OpenCV: Camera Calibration and 3D Reconstruction," [Online]. Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html.
- [7] S. C. Radu Bogdan Rusu, "3D is here: Point Cloud Library (PCL)," *IEEE International Conference on Robotics and Automation*, 2011.
- [8] "Stanford Notes on Epipolar Geometry," [Online]. Available: https://web.stanford.edu/class/cs231a/course_notes/03-epipolar-geometry.pdf.
- [9] "GStreamer," [Online]. Available: gststreamer.freedesktop.org/data/doc/gstreamer.
- [10] "Tiscamera," [Online]. Available: github.com/TheImagingSource/tiscamera.
- [11] "Unix Time," [Online]. Available: https://en.wikipedia.org/wiki/Unix_time.
- [12] S. Brahmbhatt, *Practical OpenCV*, Apress, 2013.
- [13] D. M. S. S. K. B. Pritam Prakash Shete, "A real-time stereo rectification of high definition image stream using GPU," *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2014.
- [14] "FSOCO," [Online]. Available: <https://ddavid.github.io/fsoco/>.
- [15] F. M.-N. P. F. Vincent Lepetit, "EPnP: An Accurate $O(n)$ Solution to the PnP Problem," *International Journal of Computer Vision*, 2009.
- [16] J. B. D. M. F. G. Carlos Guindel, "Automatic Extrinsic Calibration for Lidar-Stereo Vehicle Sensor Setups," *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

- [17] H. Hirschmuller, "Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.
- [18] N. D. M. Paul J. Besl, "Method for registration of 3-D shapes," *SPIE Proceedings*, vol. 1611, 1992.
- [19] "Nerian stereo camera calculator," [Online]. Available: <https://nerian.com/support/resources/calculator/>.
- [20] H. F. C. H. V. R. F. V. M. I. S. R. D. A. R. G. M. B. R. S. Miguel de la Iglesia Valls, "Design of an Autonomous Racecar: Perception, State Estimation and System Integration," *International Conference on Robotics and Automation*, 2018.
- [21] D. D. L. V. G. Ankit Dhall, "Real-time 3D Traffic Cone Detection for Autonomous Driving," *IEEE Intelligent Vehicles Symposium*, 2019.

Glossary

CNN - Convolutional Neural Network

E&C – Estimation and Control department (Driverless UPC)

FOV - Field Of View

FSD - Formula Student Driverless

GMM - Gaussian Mixture Model

GUI - Graphic User Interface

HW – Hardware department (Driverless UPC)

KNN - K Nearest Neighbours

LiDAR - Light Detection and Ranging

PAE - Projecte Avançat d'Enginyeria

PER – Perception department (Driverless UPC)

PU – Processing unit

PnP - Perspective n Point

RMSE – Root Mean Square Error

ROS - Robot Operating System

SDK - Software Development Kit

TF - Transformation Function